

A Constraint Satisfaction Approach to Operative Management of Aircraft Routing

Franco Torquati, Massimo Paltrinieri and Alberto Momigliano

Bull HN Italia
Pregnana Milanese
Milano, Italia

Abstract

This paper describes the solution to the problem of predictive and reactive aircraft routing developed in OMAR (Operative Management of Aircraft Routing), a system implemented by Bull HN in collaboration with Alitalia. The basic philosophy is to look at aircraft routing as a Constraint Satisfaction Problem (CSP) and to employ the most effective CSP techniques to solve it under the severe time constraints required by the customer.

1. Introduction

Aircraft routing is a complex task. The problem was intensively studied [Et&Ma] during the last years using analytical models supported by operations research, such as linear and dynamic programming.

However analytical models suffer from limitations that restrict their ease of use, and hence their effectiveness to solve real problems, and not just an arbitrary simplification of those problems.

As pointed out by Michie [Mi] these limitations can be summarized as:

- model complexity;
- user inscrutability;
- inability to handle uncertainty of data;
- front end complexity

These general limitations apply to our particular problem as well. Ettschneider and Mathaisel [Et&Ma] state that the problem of aircraft routing is unsolvable, at least by quantitative techniques.

Difficulties lie in the huge size of the problem and in the difficulty of expressing certain goals in quantitative terms only.

We propose to apply constraint directed reasoning and heuristic methods to develop a system for supporting human schedulers of airlines in operative management of aircraft routing. Routing can be formulated as a Constraint Satisfaction Problem (CSP): each task is a variable labelled with a set of possible values (aircraft) and the constraints are used to restrict these sets.

This paper describes the kernel of OMAR (Operative Management of Aircraft Routing), an interactive system designed for predictive and reactive routing of Alitalia fleet. Kernel's main features are:

- constraints are used to limit the search space;
- search is performed by iteratively selecting an aircraft and then assigning it to a set of consecutive flights;
- aircraft selection is driven by the first fail principle: the most constrained aircraft is scheduled first;
- a controlled form of backtracking is implemented.

The following sections define the routing problem of Alitalia fleet, its formulation as a CSP and OMAR's problem solving strategy.

2. Background on Aircraft Routing

Scheduling is the task of assigning resources to operations. Scheduling comes in two types: predictive and reactive. Predictive scheduling takes a medium term horizon and is invoked in a static and usually underconstrained environment. Reactive scheduling is a real time activity that maintains a previous plan when unexpected events occur; the environment is dynamic and usually overconstrained.

When the resources are mobile vehicles (e.g. aircraft), they describe routes through the served stations (e.g. airports). To emphasize such aspect, we will refer to this problem as the routing problem.

Routing of the Alitalia fleet is the main task of CMO (Maintenance Operation Centre) at Leonardo da Vinci International Airport, Rome. Since the Alitalia fleet covers three types of routes (domestic, European, intercontinental), a different routing plan is created daily for each of them and is maintained when unexpected events occur. The information required is distributed among paper charts, notes, on-line terminals and a large magnetic board.

Predictive routing. The main input of predictive routing is the rotation plan, a Gantt chart where bars represent flights. The rotation plan contains more information than the time table, since not only the frequency, time and airports of flights are determined, but flights are grouped in lines each assigned to a "virtual" resource, an hypothetical aircraft that could perform them in absence of technical and maintenance constraints. There are three kinds of maintenance that have to be handled:

1. Heavy maintenance lasts from several hours to days and is already planned in time and place: it is the main source of constraints in the routing of an aircraft.
2. Medium maintenance requires few hours and is executed on the dock of certain given airports. It is autonomously scheduled by the flight scheduler.
3. Light maintenance lasts only few minutes so that several of them may be executed at ground time on the same aircraft. Therefore, the only influence on the routing of an aircraft is that it reaches one of the airports in which it can be satisfied. However the distribution of light maintenance must be optimized, in order not to overload a single airport.

Medium and light maintenance, also called expiry maintenance, must be scheduled both in time and in place. Eventually technical constraints are events that prevent a specific aircraft to perform some flights.

Since the rotation plan satisfies physical and crew constraints, it is an aim of the scheduler to conform to it. Closeness to a given plan is a well defined concept that we can even measure: a routing in which maintenance activities are exactly inserted in the holes between flights belongs to the class of optimal routing; the higher the number of switches, i.e. connections between flights on different lines of the plan, the worse the solution. We may define a score according to the following formula, where the score of the optimal solution is 1:

$$\text{score} = 1 - \frac{\# \text{ switches}}{\# \text{ paired flights in the rotation plan}}$$

Routing is a trial and error process whose basic activities are:

- coupling and decoupling aircraft and flights;
- fixing or delaying start time of maintenance.

Routes are drawn on the rotation plan, performing switches to satisfy the constraints that prevent to cover the next flight of the same line. When a task that cannot be covered is detected, a previous assignment to an already scheduled one must be invalidated. The strength of the human scheduler is the comprehensive view of the rotation plan: when he switches the aircraft in process to a line that was previously scheduled, he does not involve intermediate assignments, but he easily performs non-chronological backtracking directly jumping to the line that solves the problem.

Predictive routing for the DC-9 fleet (26 aircraft, 170 flights ca.) ranges between 30 and 60 minutes.

Reactive routing. Reactive routing of the Alitalia fleet addresses the problem of revising a routing plan as unexpected events occur. The input data consists of an unexpected event, the technical constraints and a routing plan in which flights and maintenance have been previously merged and each virtual aircraft replaced by an individual aircraft. Since reactive routing is often an overconstrained problem, resource conflicts are typically solved by delaying tasks. A measure of the quality of the solution is not so straightforward as in the predictive problem, since a ranking of the candidate tasks is arbitrary and qualitative. An accepted criterion is the generation of the least total delay.

Due to the dynamic nature of the environment, response time is kept within a few minutes.

To fix the terms employed in the following sections, we will now give a more precise definition of both predictive and reactive routing of the Alitalia fleet.

Predictive problem:

Given:

a fleet, i.e. a set of n aircraft;

a rotation plan for the fleet, i.e. m ($m \leq n$) sequences of flights, with specified for each one:

- departure airport,
- arrival airport,
- departure time,
- arrival time;

a maintenance plan for the fleet, i.e. k ($k \leq n$) sequences of maintenance, with specified for each one:

- a scheduled aircraft,
- a hangar,
- start time,
- completion time;

an expiry maintenance plan for the fleet, i.e. a set of maintenance with specified for each one:

- a scheduled aircraft,
- deadline,
- a set of possible docks,
- time availability of docks,
- a set of technical constraints;

Find:

a fleet routing, i.e. n sequences of flights and maintenance such that:

- each sequence is a feasible route for an individual aircraft,
- the three plans are covered,
- airports and times of expiry maintenance are fixed,
- technical constraints are met,
- closeness to the given plans is maximal.

Reactive problem:

Given:

an unexpected event such as: aircraft unavailability at a given airport, flight departure delayed;

a routing plan as defined above;

Find:

a re-routing that most closely conforms to the given one.

From our point of view the main difference between predictive and reactive routing is that while the first problem is often underconstrained, the second is usually overconstrained.

3. Constraint Satisfaction Problems

The solution of problems as a search process driven by constraints is intensively studied in AI [Da]. Although the prototypical example of algorithm dealing with constraints (the Waltz filter [Wa]) is borrowed from image recognition, it is a widespread opinion that this approach is one of the best suited for scheduling problems. Constraints express relations between parameters, limiting the set of admissible solutions. In combinatorially complex problems constraints bound, often drastically, the number of alternatives to be explored.

A Constraint Satisfaction Problem is composed of a set of variables $V = \{X_1, \dots, X_n\}$, their related domains D_1, \dots, D_n and a set $C = \{C_1, \dots, C_k\}$ of constraints i.e. relations on the variables in V . Solving a CSP means finding every tuple from D_1, \dots, D_n which does not violate any of the constraints. Every such tuple is called a solution.

CSPs are NP-complete: since brute force search is not feasible, different types of algorithms have been developed for improving performances pruning the search space.

The first type, called tree search, assigns a value to the first variable, then to the second and so on; when a dead-end is reached backtracking is performed in non-chronological manner.

The second type, called network consistency, preprocesses the domains using constraint propagation in order to limit the search space; it can be thought as a simplification algorithm which transforms the original problem into a simpler version that has the same solutions. In some cases the resulting problem is so simple that the solutions become obvious and the original problem is solved.

The third type is called hybrid since it combines features of both the previous techniques: at each step of the search the values that cannot take part in any solution are deleted by the consistency algorithm. Much attention has been given to the problem of evaluating the optimal amount of simplification to apply at each search tree node. Though the break-even point depends on the problem involved, results in [Ha&El] and [Na] show that it is convenient to use only a very restricted form of consistency per node.

Simple heuristics, (e.g. the first fail principle) can be

applied to guide the search process [De&Pe], particularly to improve the order in which some tasks are executed, such as:

- variable assignment
- value assignment
- past variable testing
- past variable selection for backtracking.

A comparison of the different heuristics has been performed [Ha&El] on particular problems, typically the N-queens problem, where each variable constrains every other variable. A global analysis on problems in which variables are loosely constrained has not been developed yet.

4. Routing as a CSP

In this section we formally show how aircraft routing may be considered as a CSP. Since the time and place of most operations is fixed in advance, a directed acyclic network can be constructed in which each operation is a node and an arc exists between two operations if they can be processed in sequence. The problem then reduces to assigning paths in the network to resources such that all the nodes are covered.

Suppose we have the set $T = \{t_h, h=1, \dots, m\}$ of tasks to be scheduled, consisting of flights and maintenance; two tasks are said to be *connectible* (denoted $t_h \rightarrow t_k$) if the following holds:

```
connectible(Th, Tk) :-
    task_arrival_time(Th, ArrT),
    task_departure_time(Tk, DepT),
    ArrT < DepT,
    task_arrival_airport(Th, Aapt),
    task_departure_airport(Tk, Dapt),
    Aapt = Dapt.
```

This Prolog clause can be paraphrased by saying that task T_h is connectible to the task T_k if arrival time of the first comes before starting time of the second and the arrival airport of the first is equal to the departure airport of the second.

It is clear that ' \rightarrow ' is an antireflexive relation: its graph is said the task connection graph.

We say that t_h *precedes* t_k ($t_h < t_k$) if (t_h, t_k) is in the transitive closure of ' \rightarrow '. A *chain* of tasks s is a subset of T such that, for all $t_h, t_k \in s$, or $t_h < t_k$ or $t_k < t_h$. A *connected chain* or *path* is a chain such that for all $t_h, t_k \in s$, if t_k is the successor of t_h in s , then $t_h \rightarrow t_k$. The idea is that connected chains are the formalizations of the routes that a scheduled

aircraft may cover.

Next we introduce the notion of *labelling*, which describes the set of aircraft that are allowed to perform a given task as a map l from tasks to set of aircraft and show how it is calculated.

Let Acs be the set of aircraft belonging to the fleet to be scheduled and $P(Acs)$ the associated powerset. $P(Acs)$, with the inclusion relation, is a poset and this structure is inherited by the set of maps $T \rightarrow P(Acs)$.

Suppose then we have a partial map $ls(t_0): T \rightarrow P(Acs)$, which relates each aircraft at a given time t_0 to its last started task, i. e. the task still in progress or the last one executed. To find effectively the labelling in a finite number of steps we follow this procedure.

Define the ordered sequence l_0, l_1, \dots, l_n by induction as follows.

$l_0 = ls(t_0)$,
 $l_{i+1} = succ(l_i) - (l_i \cup l_{i-1} \cup \dots \cup l_0)$ where
 $succ(l_i)(t_k) =$
 $l_i(t_k) \cup s(l_i)(t_h)$ if $t_h < t_k$ and t_k is a flight
 $l_i(t_k)$ if t_k is a maintenance

If l_q is not empty and l_{q+1} is empty, then l_{q+p} is empty for all $p > 1$ and l is the disjoint union

$l = l_0 \cup l_1 \cup \dots \cup l_q$

Note that q is finite because the set of maps $T \rightarrow P(Acs)$ is finite.

From this standpoint a schedule is a member of the class of all singleton labellings, i.e. one in which each task is assigned to exactly one aircraft.

To have a CSP, we have to specify the involved constraints: in the routing problem we find two sets of constraints

1. Given two tasks t_h, t_k such that neither $t_h < t_k$ nor $t_k < t_h$, $E(l(t_h), l(t_k))$ holds iff for all $x_i \in l(t_h) \exists y \in l(t_k)$ such that $x_i = y$ and conversely.

2. Furthermore the label of a node must be included in the union of the labels of its immediate offsprings: so, $\forall t_0 \in T$ and t_1, \dots, t_r such that $t_0 \rightarrow t_i$ for $i \in 1, \dots, r$

$I(t_0, t_1, \dots, t_r)$ holds iff $l(t_0) \subseteq \bigcup_i l(t_i)$

Now we are in the position of establishing this

Proposition: if s is a singleton labelling that satisfies E and I , then $\exists a \in Acs \ I^{-1}(\{a\})$ is a connected chain.

Proof: Suppose that $\Gamma^{-1}(\{a\})$ is not a chain. Then $\exists t_h, t_k \in \Gamma^{-1}(\{a\})$ such that neither $t_h < t_k$ nor $t_k < t_h$. This implies $(\{a\}, \{a\}) \notin E(I(t_h), I(t_k))$, but $I(t_h) = I(t_k) = \{a\}$.

Suppose that $\Gamma^{-1}(\{a\})$ is not connected. Then there is a consecutive couple t_h, t_k such that $t_h < t_k$ but not $t_h \rightarrow t_k$. Consider the nonempty set $\{t_i \mid t_h \rightarrow t_i, t_i < t_k\}$: since $t_i \notin \Gamma^{-1}(\{a\})$ then $\{a\} \notin I(t_i)$ for all i .

Hence we deduce that aircraft routing may be formulated as a CSP.

As far as the labelling is concerned, constraint I is verified by construction. Constraints I and E are preserved (E is often established) through filtering. The filtering algorithm we have implemented is described below.

Let C be the connection graph; a path s may be seen as a (postfix) map linking the start node of a chain to the end one. For any subset X of nodes of C and any path s we define a subset X_s of C by the following properties:

$$1. (X_1 \cup X_2)_s = X_1s \cup X_2s$$

2. $ps = q$ if p and q are respectively the start node and the final node of the path.

From 1 and 2 we deduce

$$X_s = \{qs \mid \exists s: p < q \text{ with } p \in X\}$$

Now we consider two subsets of the nodes of C, J and L. We define S as

$$S = \{s \mid J_s \cap L \text{ not empty}\}.$$

Every node in an element of S has the feature of belonging to some path connecting J to L. In our model L is a set of maintenance or distinguished flights, J the set of the last started tasks of the aircraft that must perform L. The main property of a maintenance is that its label is a singleton aircraft and, in order to satisfy relations E and I, that reduces other labels of the network. More precisely aircraft associated to L must be deleted from the labels of the nodes in the complement of S.

This is the target of the refinement algorithm we named 'trim_refine' after the definition of trim automata [Ei].

5. System Architecture

In this section we sketch the system architecture performing a (simplified) guided tour through OMAR's basic operations cycle in order to make clear how the schedule is worked out, focusing on the DC-9 fleet, one of the largest in Alitalia.

At the start of the session the state of the fleet and the information on the tasks to be scheduled are loaded from the Alitalia database.

At this stage the system tests whether at each moment of the schedule there are enough available resources (aircraft) for the planned tasks. It is clear that this is a necessary (but not sufficient) condition for the existence of a schedule: if a rotation is feasible, the number of available resources must always be greater than or equal to zero.

An algorithm linear in the number of tasks checks this condition: if it holds, the system goes on, otherwise the plans are modified either interactively by the user or automatically by the system following a fifo strategy: the next starting task is delayed until a resource becomes available.

Then the system enters its second level, building the derived data structures on which the consistency techniques are to be applied: the task connection graph and the task labelling.

Yet the user may enter a phase of pre-scheduling where he can impose additional constraints such as:

- establishing a unique link between sequences of tasks;
- setting (deleting) a single ac on (from) a task;
- scheduling an ac on a distinguished path.

The 'trim_refine' algorithm shrinks the labels so that most dead-ends are avoided and expiry maintenance requirements are implicitly satisfied: that means that aircraft planned for the latter tasks are excluded by those routes which do not lead to the set of airports where maintenance is not possible.

If the network is not found consistent, no complete routing exists, so the control comes back to the human scheduler who, based on his own unique experience, relaxes the relevant constraints. It is our opinion that this kind of expertise cannot be adequately simulated by a computer, since the knowledge required to recognize the causes of an inconsistent situation and to suggest a recovery solution is too extended and fuzzy. If, on the other hand, everything is successful, the system is ready to schedule.

Before the routing process the aircraft are sorted in decreasing order according to the number of occurrences inside the labelling; the idea is that the aircraft coming first in this order are the most constrained ones, since they have a smaller number of task on which they can be enrounted.

Routes are then created according to such order by the procedure sketched below:

```
route_gen([Ac|Acs],[Ac-Path|Routing]):-
```

```
    pathgen(Ac,Path),
```

```
    !,
```

```
    route_gen(Acs,Routing).
```

```
route_gen([],[]).
```

```
path_gen(Ac,Path):-
```

```
    last_started(Ac,Task),
```

```
    path_gen(Ac,Task,Path).
```

```
path_gen(Ac,Task,[NextTask|Path]):-
```

```
    select(Ac,Task,NextTask),
```

```
    !,
```

```
    path_gen(Ac,NextTask,Path).
```

```
path_gen(_Ac,_Task,[]).
```

```
select(Ac,Task,NextTask) :-
```

```
    method(Ac,Task,Methods),
```

```
    member(Method,Methods),
```

```
    propose(Method,Ac,Task,NextTask).
```

Given an aircraft *Ac*, *pathgen/2* returns without backtracking (note the use of *!*) a path *Path*. *Path* is iteratively generated by *select/3*, a procedure that, given an *Ac* and a *Task*, selects *NextTask* according to *methods* such as: choose the next task on the same line, choose the closer task, make a switch, etc. These methods are tried in fixed sequences, called *strategies*, until one succeeds. Different strategies allow the user to obtain schedules satisfying various kind of requirements.

A dynamic strategy is also present which permutes the ordering of the methods during the search, aiming to employ greedy methods just when resource availability is considered critical.

Experimental results have evidenced that label refinement and dynamic re-ordering of aircraft after path generation is not cost effective. In this sense OMAR does not implement an hybrid tree search/network consistency algorithm. Moreover the system performs a very limited amount of backtracking: different choices are considered only during the coupling of a task with one of its direct offspring. Yet paths cannot be invalidated. The latter aircraft, which are less constrained, fill the potential gaps left by the former ones, trying to secure in this way the coverage of all the tasks.

This approach does not ensure to find a complete solution even if one exists; nevertheless we maintain that

there are several good reasons not to allow an unrestricted form of backtracking:

- in ac routing, since the search is strongly guided by the connection graph, most dead-ends are avoided by filtering during the pre-search phase;
- in case of failure, unrestricted backtracking is not practical since, due to the number and the interdependency of the involved constraints, the time necessary for a solution is unpredictable and memory requirements often unacceptable;
- experimental evidence has shown that, whenever route generation procedure fails, the problem is overconstrained and a solution is not attainable.

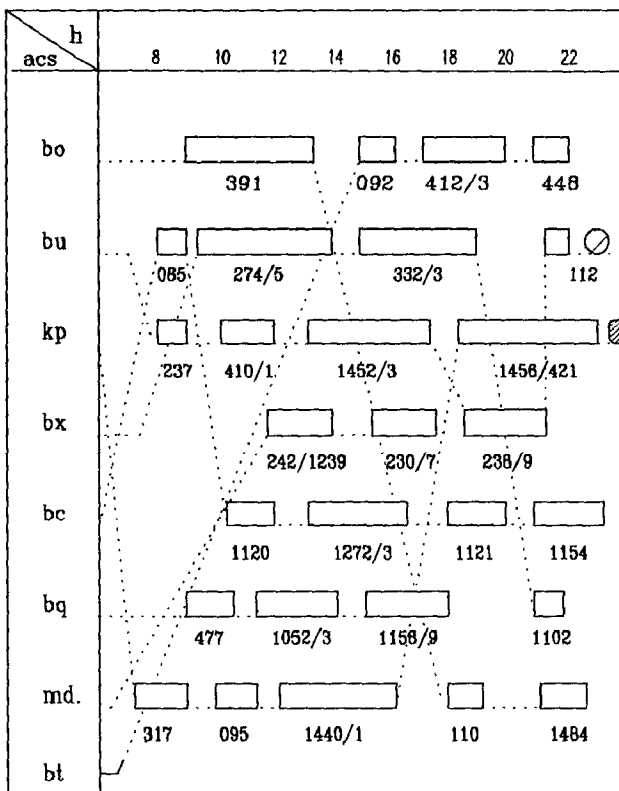
In such a case, the system gives some guidance and releases the control to the user. In our experience, after the relevant modifications have been performed, another run of the scheduler is generally sufficient to achieve a satisfying solution.

Scheduling example. Suppose we have the portion of the rotation plan for DC-9 fleet for a typical working day (5/3/1989) shown below. As already said, every line corresponds to a set of flights that must be assigned to a individual aircraft.

| h | acs | h | | | | | | |
|---|-----|---------|----------|---------|-------|----------|-----|---------|
| | | 8 | 10 | 12 | 14 | 16 | 18 | 20 22 |
| 1 | | sto | lin | fco | lin | gva | lin | dus |
| | | | | | | | | |
| 2 | | 391 | | 092 | 442/3 | 448 | | |
| | | lin fco | bru | fco | par | fco | | aho |
| 3 | | 085 | 274/5 | | 332/3 | | | 112 |
| | | aho | fco gva | fco | par | fco | fra | fcp |
| 4 | | 237 | 410/1 | 1452/3 | | 1458/421 | | |
| | | fco vrn | fco blq | fco blq | fco | lin | fco | |
| 5 | | 1156/1 | 242/1239 | 230/7 | 238/9 | | | |
| | | vrn | fco psa | lin | bru | lin | psa | lin vrn |
| 6 | | 1155 | 1120 | 1272/3 | 1121 | 1154 | | |
| | | muc | fco goa | fco vrn | fco | psa | | |
| 7 | | 477 | 1052/3 | 1158/9 | 1102 | | | |
| | | par | lin | fco fra | fco | lin | ham | |
| | | 317 | 095 | 1440/1 | 110 | 1484 | | |

- Furthermore the following table of constraints is given (no light maintenance is considered for clarity reason):
- *bo* must execute flight az110;
- *bu* has a medium maintenance *isa* lasting two hours and executable only at *psa* and *aho* airport;
- *kp* has a heavy maintenance *isb* at *fco* from 5/23/30 to 7/23/30;
- *bx* must remain overnight at *fco*; in addition it has a technical constraint (being not provided with automatic power unit) that excludes it from going to *blq* airport.

Eventually we stipulate that switches will not be permitted unless at *fco* airport. We now follow OMAR scheduling this portion of the fleet. Since *bo* is the most constrained one, it is scheduled first; from line 1 it jumps to line 7 as required and then stops; *bu* arrived at 1453 cannot go further on line 3 because it would reach *fco* where it cannot perform *isa*: therefore it greedely switches to line 4 and then back to 2 since flight 113 ends at one of the airports required by *isa*; *kp* follows part of line 7, then after 1441 it jumps to line 3 to do the *isb*. Now it is the turn of *bx* that shall not follow its line not to sink in *blq*: so it sticks to line 2 and quietly switches to 6, since the terminal task of the latter line is already covered by *bu*. Analogously the remaining aircraft are driven to cover the gaps left by the more constrained ones, as is shown below.



6. Conclusions

The problem of aircraft routing is very complicated and no perfect solution has been achieved. Consequently all models are heuristic and work is now concentrating on the systematic interaction between human and computer. OMAR's approach to the problem appears very promising. System response time is satisfactory: once the derived structures have been computed, routing comes up in 15 seconds. In the tests supplied by Alitalia, OMAR's solutions can be compared with the one shown by a senior scheduler. OMAR's kernel is at the moment composed of nearly 20,000 lines of Quintus Prolog code running on a Bull X-20 minicomputer.

Acknowledgements

Several meetings were necessary to achieve a deep knowledge of the problem and of the solution techniques employed by the human schedulers. This was made possible thanks to the understanding and collaboration shown at CMO. We are also indebted to the members of the ROI (Operations Research) group at Centro Elaborazione Dati Alitalia who focused our attention on the problem and continue to contribute to OMAR project, especially in the database definition and in the implementation of the connection with Alitalia Information System. Special thanks to our colleagues in Expert System Division at Pregnana, for providing the excellent graphic interface to OMAR.

References

- [Da] Davis E., "Constraint Propagation with Interval Labels", *Artificial Intelligence*, 32, 1987, 281-331.
- [De&Pe] Dectcher R. & Pearl J., "Network-Based Heuristics for Constraint Satisfaction Problems", *Artificial Intelligence*, 34, 1988, 1-38.
- [Ei] Eilenberg S., "Automata, Languages and Machines", Academic Press, New York, 1977.
- [Et&Ma] Etschmeier M.M. & Mathaisel D.F.X., "Aircraft Scheduling: the State of the Art", XXIV AGIFORS Symposium, Strassbourg, 1984.
- [Ha&El] Haralick R.M. & Elliot G.L., "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", *Artificial Intelligence*, 14, 1980.
- [Mi] Michie D., "Game Playing Programs and the Conceptual Interface", *Sigart Newsletter*, April 1982.
- [Na] Nadel B.A., "Tree Search and Arc Consistency in Constraint Satisfaction Problems", in Kanal & Kumar (eds), *Search in Artificial Intelligence*, Springer-Verlag, 1988.